HIERARCHICAL SEGMENTATION OF FOVEAL IMAGES

Dr. Cesar Bandera and Andrew Izatt

bandera @amherst.com izatt@amherst.com

Amherst Systems, Inc.
30 Wilson Road
Buffalo, New York 14221
Tel: (716)631-0610

Abstract

This paper describes a technique for the segmentation of foveal images and other foveal retinotopic feature maps. The technique is implemented hierarchically in the small subset of an image pyramid, called the foveal polygon, supported by foveal imaging. Unlike binary or single spot detectors, this technique segments into multiple classes; the number of classes is determined by the image itself, and a maximum within-segment feature variance threshold. The technique represents segments, which need not be retinotopically contiguous, as subtrees in the polygon. These subtrees are used for efficient segment analysis and manipulation; the subtree nodes representing a single compact region are quickly identified and labeled, and statistics are efficiently computed at all levels of the hierarchy, including on segments and compact regions.

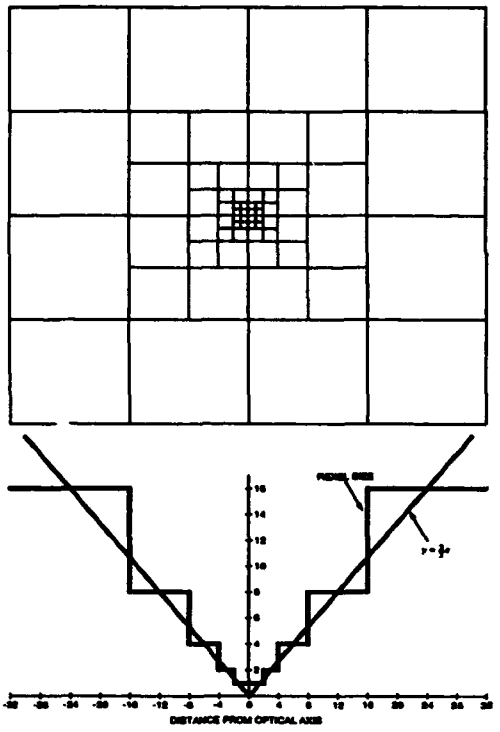
Region segmentation is a fundamental component of almost every machine vision system. Hierarchical segmentation on the pyramid has been demonstrated to yield better results than conventional 2-D segmentation techniques [Hird89], and has been shown to quickly converge to a stable solution [Cibulskis84]. This paper examines the hierarchical segmentation techniques developed for the foveal polygon. The basic segmentation technique was adapted from the multiclass pyramidal techniques developed by Burt and Rosenfeld [Burt81] [Hong82]. Several extensions have improved segmentation results, yielding multiple homogeneous, compact regions within segments. It has also been extended to work with foveal data.

I. INTRODUCTION

In contrast to the uniform acuity of conventional machine vision, virtually all advanced biological vision systems sample the scene in a space variant fashion. Mammalian retinal acuity varies by several orders of magnitude within the field of view (FOV) [Levine85]. The region of the retina with notably high acuity, called the fovea, is a small percentage of the overall FOV centered at the optical axis. In human vision, the fovea accounts for 1° out of a total FOV of 200°, or 0.0025% for the FOV area. The wide FOV, supported by lower peripheral acuity, and the high acuity fovea impose a much smaller data set than supporting the entire FOV uniformly at high acuity. Inherent with space variant sampling is the context sensitive articulation of the sensor's optical axis, whereby the fovea is aligned with relevant features in the scene. These features can be targets (e.g., predators or prey), or classification features on the targets themselves. Space variant sampling and intelligent gaze control are collectively called foveal vision.

Figure 1 illustrates a lattice implementing what we call the exponential foveal geometry. The sensor elements are called resolution cells, or *rexels*, to distinguish them from the uniformly sized sensor elements, or *pixels*, of conventional imaging. The exponential lattice contains a 4×4 fovea array of uniformly sized rexels (the size of each fovea rexel can be normalized to 1×1). The fovea is surrounded by a ring of rexels each of size 2×2, which is surrounded by a ring of rexels each of size 4×4, which is surrounded by a ring of rexels each of size 8×8, and so forth.

The exponential lattice features an inverse linear acuity roll-off. Rexel size is proportional to the L_∞ distance from its center to the lattice center, except at the uniform fovea. Localized acuity is inversely proportional to rexel size, and is thus inversely proportional to distance from lattice center. Note that in Figure 1 each ring consists of the same number of rexels (twelve), and that adding an additional ring increases the field-of-view (FOV) by a factor of four. A wide FOV can thus be attained with little cost in data and processing bandwidth.



The fovea and first four rings are shown.

Figure 1 Root Exponential Foveal Lattice

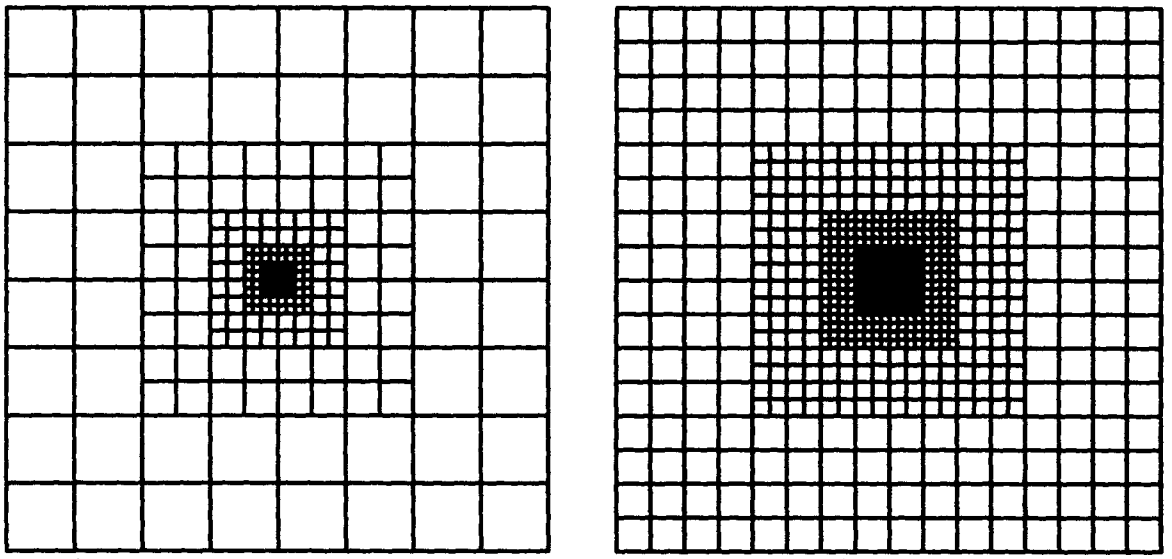
The lattice in Figure 1 is also referred to as the root lattice because from it many other lattices can because from it many other lattices can be derived with the same properties (perfect tiling of square rexels related in size by power of two) but with flatter acuity profiles. A flatter acuity profile will typically be required because that of the root lattice is very steep and produces a peripheral acuity which is too coarse to be of much value in most imaging applications. The derived lattices, called subdivided exponential lattices, are obtained by uniformly subdividing each rexel in the root lattice by a subdivision factor d, and scaling the geometry upward by the same factor d to preserve the maximum acuity (i.e., maintain the size of fovea rexels normalized to 1×1). The subdivided lattice

A-1

has a fovea of size $4d \times 4d$, and d rings of a given rexel size (the collection of d rings with the same sized rexels is called a major ring), and an acuity profile given by

$$\text{acuity} = \frac{1}{\text{rexel size}} \cong \frac{3}{2} \cdot \frac{d}{L_\infty \text{ distance from lattice center}} \qquad \text{(eq. 1)}$$

Two subdivided exponential foveal lattices are shown in Figure 2.



a. Four major rings about the fovea, d=2 b. Three major rings about the fovea, d=4

Figure 2 Subdivided Exponential Foveal Lattices

A class of hierarchical architectures for the representation and multiprocessing of foveal data has been developed by Amherst Systems and the University of Buffalo to translate the efficient use of sensor signal bandwidth into quick system response time. The architecture, called the foveal polygon, resembles the hierarchical processing in the vertebrate visual cortex. Multiacuity data is processed homogeneously across acuity levels (SIMD operation is supported). The polygon also resembles the image pyramid architectures used in current machine vision. In fact, the pyramid is a special case of the polygon when sensor acuity is uniform. Maximum acuity sensor data (fovea measurements) is supported at the base, and minimum acuity data (peripheral measurements) at the polygon waist. This three dimensional allocation of rexels in the polygon, called the rexel manifold (dark cells in Fig. 3), stores multiacuity data more efficiently than a 2-D array.

The polygon is a "tube" of relatively small uniformly sized arrays. Each level is identical to the one below (except for the top crown above the waist). Consequently, fine grain polygons are more feasible than fine grain pyramids. The height of the polygon is determined primarily by the number of rings in the sensor geometry. Each ring contributes to the level associated with the ring's resolution, with the fovea rexels mapping to the base, and peripheral rexels mapping to the polygon waist. The area of a polygon level up to the waist is $4d \times 4d$, where d is the lattice subdivision factor, so the sharper the acuity profile, the narrower the polygon. Note that in Figure 3 all the levels up to the waist of a given polygon are of the same size, trading off FOV for resolution.

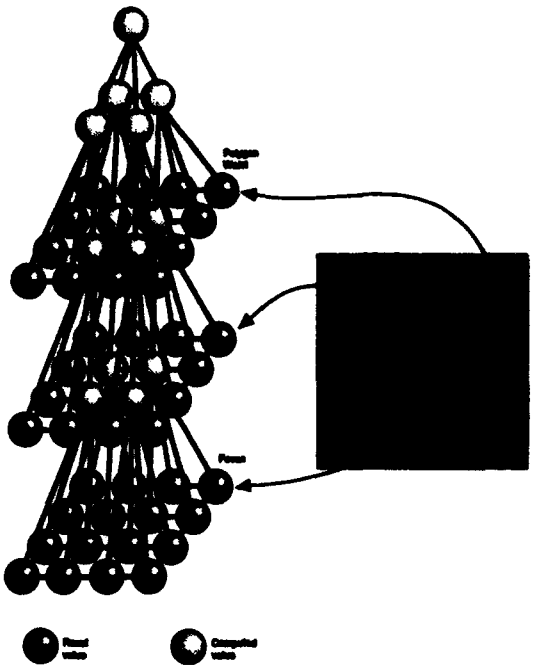


Figure 3 The Foveal Polygon Data Structure and Processing Hierarchy

It is in the development of early vision functions where our work differs most from foveal vision efforts using polar symmetric imaging. This is to be expected, since the topology of efficient early vision processing should match the sampling topology. An advantage of the Cartesian symmetric lattice is that the resulting data structure and processing engine support a wide collection of vision algorithms developed for the image pyramid. System development is accelerated by adapting pyramid algorithms to the polygon, as opposed to developing new algorithms on some lesser understood topology.

II. PYRAMIDAL SEGMENTATION

Region segmentation is a fundamental component of almost every machine vision system. Hierarchical segmentation on the pyramid has been demonstrated to yield better results than conventional 2-D segmentation techniques [Hird89], and has been shown to quickly converge to a stable solution [Cibulskis84]. Many pyramidal topologies have been proposed, including polar configurations [Stout86] [Porat88] [Peleg87]. Most algorithm development has focused on two particular types of pyramids, typically called 4-to-1 and 16-to-4. In both topologies, the linear dimensions of each level in the pyramid are half that of the level underneath. A pyramid formed from a $2^N \times 2^N$ pixel base image has $N+1$ levels of size $2^N \times 2^N$, $2^{N-1} \times 2^{N-1}$,...,1×1, and a total of $\frac{4}{3}2^{2N} - \frac{1}{3}$ cells. The difference between the two topologies lies in the method employed to compute the values of the cells above the base. In the 4-to-1 pyramid, the value of a cell above the base (a "father" cell) is simply the average of the 2×2 ("son") cells underneath. Here, *underneath* means the cells of the level below which compose the same local field-of-view of the father cell. As the name implies, the value of a father cell in the 16-to-4 pyramid is the average of the 4×4 cells underneath. Each son cell contributes to the value of four father cells. Two horizontally or vertically adjacent father cells share two son cells, i.e., their receptive fields on the level below overlap by 50%. For this reason, the 16-to-4 topology is called an overlapping pyramid.

The hierarchical segmentation technique adapted for foveal vision is adapted from the multiclass pyramidal techniques developed by Burt and Rosenfeld [Burt81] [Hong82]. The term multiclass means it segments the input into a controllable number of segments, or classes, as opposed to a simple binarizer which segments its input into two classes. The segmentation technique is applicable to gray value data, color, motion, or any other retinotopic feature that can be represented in coarser resolution through an averaging process. The basic technique for the 16-to-4 pyramid consists of steps given in Table 1, and is based on the bottom-up (fine-to-coarse) process whereby a cell chooses and links to one father out of four possible fathers permitted by the 16-to-4 topology, based on value similarity. Father values are then computed from the sons linked to it.

This segmentation algorithm simultaneously performs image smoothing and edge enhancement. Similar valued cells are grouped into a common subtree because they link to father cells which are also similar in value. As the algorithm iterates, the values of father cells are computed from a more homogeneous subset of pixels (i.e., subtree leaves). Likewise, the lateral inhibition resisting the linking of a son cell to a dissimilar father cell differentiates between father cell values and the corresponding subtrees. Thus, as the algorithm iterates, the values of different subtrees become more different. Similarities and differences between pixels and cells are simultaneously exploited, leading to a quick and guaranteed convergence of the algorithm to stable subtrees. Because the algorithm is driven by the similarity and difference between regions, as opposed to their absolute feature values, it is not affected by feature scaling, such as uniform changes in illumination or contrast.

Table 1 Basic Pyramidal Hierarchical Segmentation Algorithm

1. Generate (initialize) a 16-to-4 pyramid with cell extend border. The value of each father cell is set to the average of its 4×4 son cells. Each cell below the top contributes to the value of four father cells. A pyramid with a $2^N \times 2^N$ base has $N+1$ levels (base is level 0, apex is level N).
2. For all levels below the top, link sons to fathers according to their similarity. Each cell below the top has four candidate father cells. For each cell, its most similarly valued father is identified, and a link is established between the son cell and father cell. A son is allowed to link to only one father cell. Each son has one father, but a father can have from zero to 16 sons (if a father has no sons, its value is selected at random from its candidate 16 sons). If the links are the same as those computed in the previous iteration, the pyramid has stabilized and the algorithm proceeds to step 4. Otherwise, it continues to step 3.
3. Regenerate the pyramid using the new links. The value of each father cell is recomputed using the average of the values of the sons linked to it. This exhibits a lateral inhibition and non-linear high-pass filtering which accentuates differences between father cell values, while grouping similarly valued son cells. The algorithm then returns to step 2.
4. Generate the segment class values (top-down pass). The pyramid now consists of a tree structure with one root starting at level N and with all the base level image pixels as the leaves. The number of classes is determined by selecting a level L in the pyramid at which the tree is "cut" leaving a group of 4^{N-L} subtrees. A subtree represents a segmentation class. The value of a class is computed as the average of all its subtree leaves. This value is obtained directly from the root of the subtree at level L and is propagated downward by assigning each cell below level L the value of its father, starting with father-son pairs at level L-1, then L-2, and so forth. Because two or more cells at the level L can have the same value, there may be less than 4^{N-L} distinct segmentation values.

III. VARIATIONS TO HIERARCHICAL SEGMENTATION

Several variations of the algorithm in Table 1 can improve segmentation and reduce processing [Burt81] [Hong82] [Bandera91]. First, pyramid initialization is changed from 16-to-4 linking to 4-to-1 linking (Figure 4). This improves the segmentation of small regions by preserving scene details at higher levels and by increasing the chances of an appropriate father to exist for a small region of uniquely valued son cells, which might otherwise be washed out by 16-to-4 averaging. It also circumvents the gray level biasing and computational overhead of border padding during pyramid initialization (all the sons of a father in a 4-to-1 pyramid exist).

Another variation to the algorithm is the exclusion of "sonless fathers." In the initial algorithm, the value of a father cell to which no son cells from the level underneath are linked was set to the value of a randomly selected candidate son. Such a father cell could add artifacts to the segmentation process by influencing the value of a father cell above with inappropriate data. The arbitrary selection of a son cell implies that a son contributes to more than one father, weakening the algorithm's lateral inhibition, and doubly counting some scene feature.

Another variation is to weigh son values by their area when computing the value of a father. The area is the number of leaves on the subtree below that cell. The areas of the cells at k=0 are all set to one, and the cells above are initialized with an area of zero. During pyramid regeneration after cell relinking (step 3), the area of a cell above the base is simply the sum of the areas of its sons.

Weighted averaging, whereby the value vector of a son is weighted proportionally by its area, better preserves the integrity of region values as subregions are grouped. The old equation for the value of a father is given by

$$f = \frac{\sum_{i \in S} \mu_i}{\sum_{i \in S} 1} \qquad \text{(eq. 2)}$$

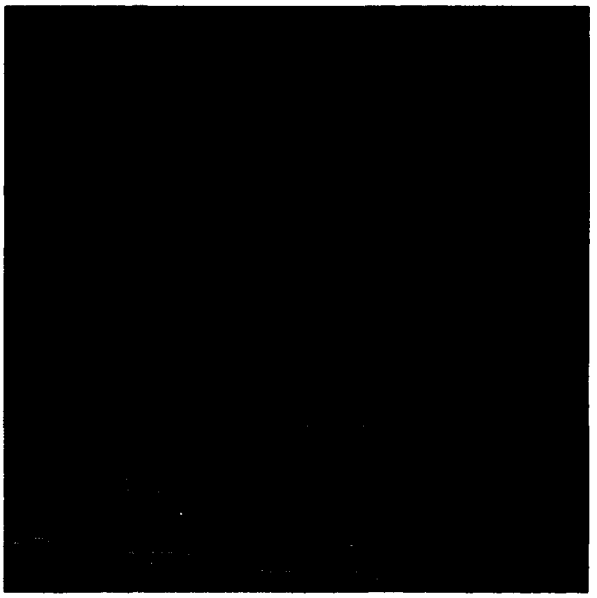
where f is the father value vector, μ is the value vector of a linked son, and S is the set of indices of sons linked to that father. The new equation for the value of a father is

$$f = \frac{\sum_{i \in S} \mu_i a_i}{\sum_{i \in S} a_i} \qquad \text{(eq. 3)}$$

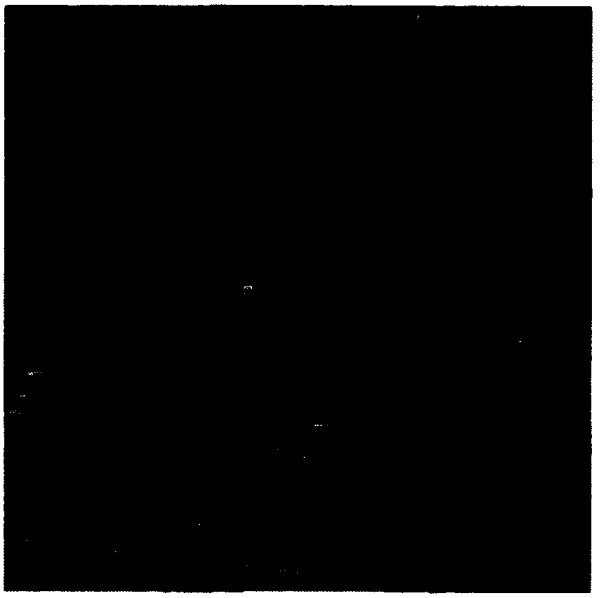
where a is the area of a linked son. A son without children has an area of zero, and has no influence on its father's value.

Another variation is to use torodial levels. This wrap around path permits the same 16-to-4 linking structure to be used homogeneously throughout the pyramid, and does away with level padding. It also provides a shortcut in which regions with the same properties but spaced far apart in the image, can link together.

a. Input image.



b. 16-to-4 initialization.



c. 4-to-1 initialization.

Figure 4 Segmentation with Different Initialization Techniques

Another variation is an alternate father-son similarity measure is $\Delta(D+s)$, where Δ is the difference in value, D is the distance between cell centers, and s is a sensitivity factor which, for larger values, attenuates the effect of cell distance in the measure.

The algorithm execution loop can be restructured to reduce the computational requirements of the algorithm when executed sequentially on a uniprocessor or two dimensional processor array, the latter configuration having been often used to emulate a pyramid [Burt88] [Cantoni86] [Cantoni88] [Stout88]. The linking to and regeneration of father cells (Steps 2 and 3 in Table 1) at a given level

does not affect the linking and father regeneration of the levels underneath. The steps of the algorithm are repeated bottom-up until the single top-down pass is performed as the last step. Consequently, each level can be processed sequentially starting with the base. The links between the base and level 1 are iterated and level 1 is regenerated until stabilization is achieved. The algorithm then proceeds with level 2 to level 3 linking, and so forth. Processing a level after the levels below are stabilized (known as level sequential segmentation) eliminates the reprocessing in pyramidal iterative segmentation (Table 1) of the entire structure due to localized relinking of cells underneath.

Whereas level sequential segmentation is faster in sequential implementations of pyramids using 0-D (uniprocessor), 1-D (vector processor) and 2-D (array processor) architectures, it is not necessarily recommended for true 3-D implementations of pyramid or polygon architectures because some degree of stable linking can occur concurrently at higher levels prior to the stabilization of lower levels.

An interesting feature of the segmentation algorithm is the evolution of the levels above the base. Initially, these levels are progressively lowpass filtered and decimated versions of the input image. After stabilization of the links, however, the generation of the father cells are not necessarily shift invariant; a father cell at level k may be the average of $(2^k+2)^2$ pixels, or it may be linked to only a single pixel. Indeed, the pyramid levels after segmentation stabilization more closely resemble a decimated median filtering of the input image.

IV. POLYGONAL SEGMENTATION

A. Basic Polygonal Segmentation

A key difference between algorithms on the pyramid and their adaptation on the polygon is the systolic data flow. The driving data in the pyramid exists at the base level, and the integrity of this data is preserved in all bottom-up algorithms. For example, pyramid initialization and segmentation does not alter the pixel values. However, the driving data in the polygon is in the rexel manifold, and appears at all the levels up to the waist. Care must be taken when adapting pyramid techniques to the polygon that the same causality in systolic data flow is preserved, even though the driving data in the polygon is not as segregated from the rest of the data structure. For example, the bottom-up averaging process which initializes the 4-to-1 polygon uses rexel values to compute the values of cells in the interior of higher levels without changing the rexel values. Likewise, the relinking step in the segmentation process should preserve the rexel values as the process driver.

The 4-to-1 polygon can be used to initialize the data structure for segmentation. However, an overlapping cell hierarchy is required for the segmentation so that a cell has several fathers from which to choose and adaptively link. A 16-to-4 polygon architecture with top-left justified linking is presented in [Bandera91]. This polygon segmentation technique uses torodial level above the waist, and a center justified overlapping architecture described below.

A $4d \times 4d$ polygon level can be divided into a central $2d \times 2d$ region containing values computed from the level below during 4-to-1 initialization, and an outer ring d cells wide containing rexel values (this being a cross-slice of the rexel manifold). Each cell not on the outer edge of a polygon level is a candidate son to four fathers in the computed value region of the level above. Because these fathers support computed rather than measured (rexel) values, the process of linking to them will not alter raw data. However, if the same linking structure is homogeneously applied throughout the polygon, then cells on the outer edge of their level can link to father cells which are on the rexel manifold (specifically on the inner ring of the manifold).

The linking of sons on the outer edge of a level to fathers on the interior manifold of the level above represents the grouping of adjacent scene regions across acuity boundaries of the foveal lattice. These links are just as valid as grouping regions within a given acuity, and must be permitted to exist. Figure 5 illustrates the coverage of two adjacent polygon levels, and son-to-father links in the 16-to-4 polygon. Note that in Figure 5 the four candidate fathers of any son are those whose centers are closest to the center of the son cell.

One technique which preserves the integrity of both measured data and links across acuity boundaries uses "implicit son cells" to support the rexel data assigned by the 4-to-1 pyramid while permitting the value of a father cell on the manifold to be influenced by son cells that have linked to it. Each inner manifold cell above the polygon base will be assigned four implicit sons, each containing the rexel value, an area equal to one fourth that of the rexel, and a link to the inner peripheral father.

Implicit sons will be static; they are not fathers of any other cells, their driving value (e.g., measured color value) is never altered, and they are always linked to the father whose rexel value they contain. Thus, if no actual sons from the level below link with this father, its value will be the same as that assigned by the 4-to-1 polygon. On the other hand, if an actual son does link to the father, the father's value upon bottom-up recomputation will be an average of the rexel value and the linked son's value.

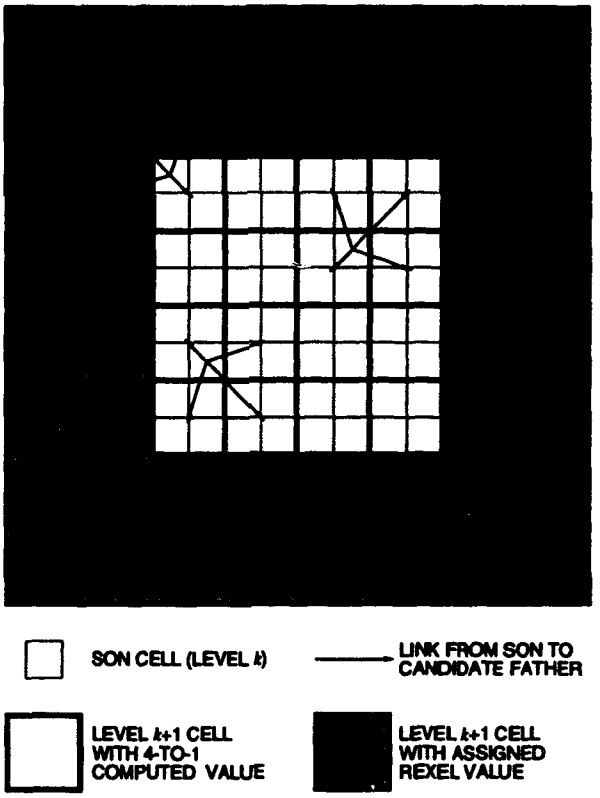


Figure 5 16-4 Linking in a d=2 Polygon

The bottom-up steps of the polygon segmentation process are now defined: 4-to-1 initialization of the polygon with direct assignment of rexels to polygon cells, followed by the 16-to-4 linking of sons to fathers using implicit sons to represent the rexel values. As with pyramidal segmentation, the bottom-up linking can be performed either as a polygon iterative (corresponding to the pyramidal

iterative approach) or a level sequential process. The polygon iterative segmentation algorithm is summarized in Table 2, and the polygon level iterative algorithm is summarized in Table 3.

Table 2 Polygon Iterative Segmentation Algorithm

1. Generate (initialize) a 4-to-1 polygon. The value of each father cell is set to the average of its 2×2 son cells. Each cell below the top contributes to the value of one father cell. The area of each cell on the manifold is assigned the area of the corresponding rexel. Cells not on the manifold are assigned an area of zero.
2. For all levels below the top, link sons to fathers according to their similarity. Each cell below the top has up to four father cells. For each cell, its most similarly valued father is identified, and a link is established between the son and father cell. A son is allowed to link to only one father cell. If none of the links computed in the previous iteration are changed, the polygon has stabilized and the algorithm proceeds to step 4. Otherwise, it continues to step 3.
3. Regenerate the polygon using the new links. The value of each father cell is recomputed using the area weighted average of the sons linked to it (including implicit sons). The algorithm then returns to step 2.
4. Generate the segment class values (top-down pass). Propagate down each subtree the value of its root at level L. The segmentation results appear at the rexel manifold.

Table 3 Level Sequential Polygon Segmentation Algorithm

1. Generate (initialize) a 4-to-1 polygon. The gray value of each father cell is set to the average of its 2×2 son cells. Each cell below the top contributes to the gray value of one father cell. Working level k=0. The area of each cell on the manifold is assigned the area of the corresponding rexel. Cells not on the manifold are assigned an area of zero.
2. Link the cells of level k to fathers according to their similarity. Each cell below the top has up to four father cells. For each cell, its most similarly valued father is identified, and a link is established between the son and father cell.

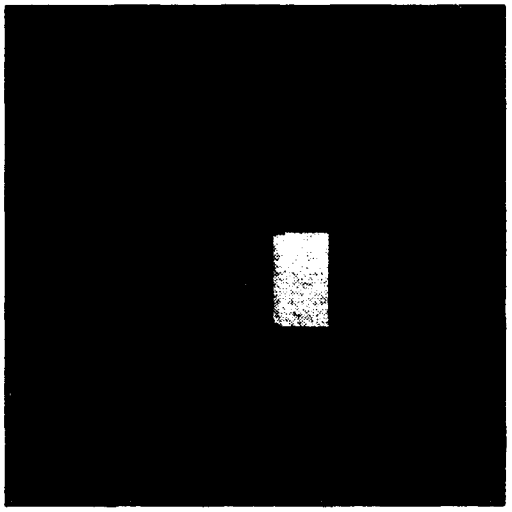
2a. If none of the links computed for level k in the previous iteration are changed, then the polygon level k is stabilized, and the algorithm repeats step 2 with the next level selected for processing ($k \leftarrow k+1$). If k is the top level of the polygon, all levels of sons have been linked and stabilized, and the algorithm proceeds to step 4.

3. Regenerate level $k+1$ using the new links. The value of each father cell is recomputed using the area weighted average of the sons linked to it (including implicit sons). The algorithm then returns to step 2.
4. Generate the segment class values (top-down pass). Propagate down each subtree the value of its root at level L. The segmentation results appear at the rexel manifold.

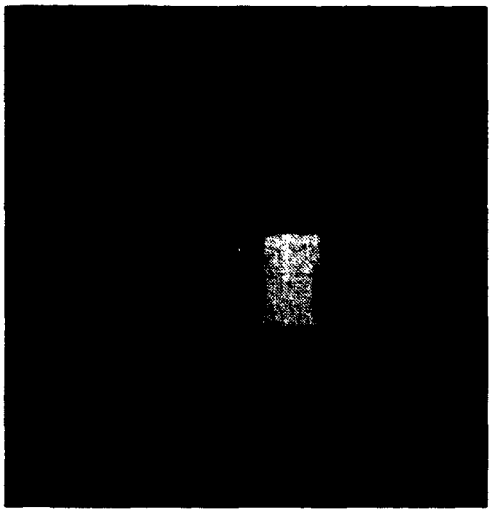
The segmentation output of the pyramid is taken to be the segmentation values at the leaves of the various subtrees in the hierarchical structure. The same approach is followed for the polygon. One difference between the two approaches is that while the leaves of the pyramid all exist at the base level, the polygon leaves exist in the rexel manifold which extends at all levels up to the waist. It could be argued that cells on the inner rings of the manifold are not technically subtree leaves because they can have cells linking to them. However, the output of a segmentation is a labeled image frame, and in the foveal setting the frame is stored in the manifold.

The final top-down propagation of segmentation values is a straightforward extension of the pyramid step. Beginning at some level L selected to provide the desired number of segments, the cell values

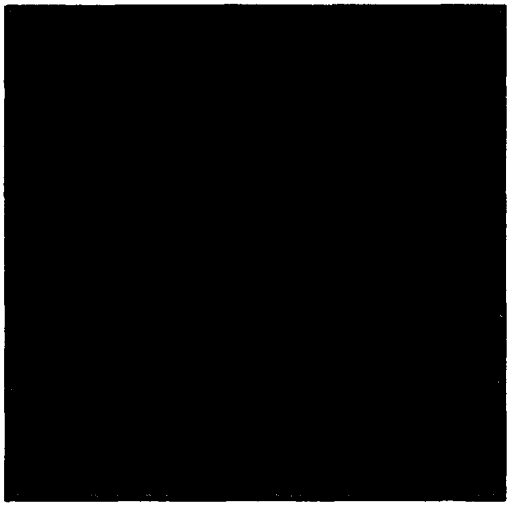
are propagated down to the linked sons, grandsons, etc. This top-down wave of level L values effects all cells, including those assigned rexels and sonless fathers. Images 6c and 6d show the results of the level sequential segmentation process on input image 6a. The displayed color value is the segmentation value which has been propagated down from the root level L through the polygon. The number of segmentation values is determined by the selection of the root segmentation level L, where the number of segments is equal to 4^{N-L} Figure 6c has a total of 4 segmentation values. Figure 6d has a total of 16 segmentation values.



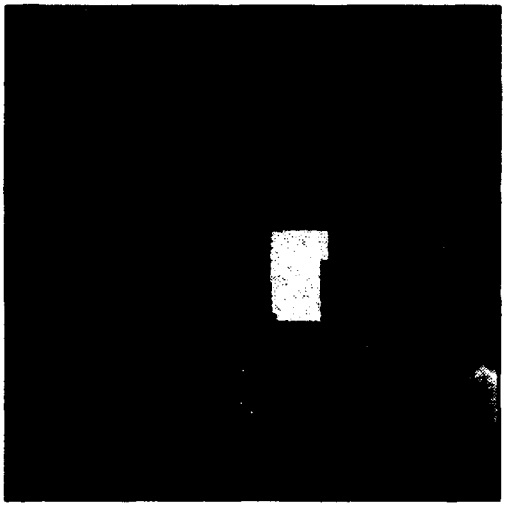
a. Uniform acuity input image.



b. Rexel image; 3 rings, d=8, ???? rexels



c. Segmentation results at L=7.



d. Segmentation results at L=6.

Figure 6 Segmentation Variations Based on the Root Segment Level.

A drawback of this algorithm is the imposition of a fixed number of classes via the selection of a root segmentation level L, which results in the loss of valuable segmentation information. Comparison of segmentation results in Figures 6c and 6d illustrates the importance of selecting the root segmentation level L. If L is too high, distinct homogeneous segments found at level L-1 may be forced to merge into a heterogeneous segment at level L, obscuring relevant segmentation results which are available at level L-1. This is illustrated clearly in Figure 6c where distinct segments found in Figure 6d are forced to merge with the background and other targets. If the root segmentation level L is selected too low, scene objects are decomposed into many segments, defeating the very purpose of segmentation, which is to group pixels or rexels into meaningful classes. This complicates

subsequent mid-level vision processing, like object detection, by necessitating additional segment grouping/decomposing. A more flexible approach is required which supports the determination of an optimal number of classes as a function of image properties, and not just hierarchy topology.

B. Image Sensitive Class Count

By definition, an image segment is a group of pixels or rexels, not necessarily contiguous, with values that are similar, to within some similarity threshold. While similarity is used to drive segmentation, the algorithms in Tables 1, 2, and 3 do not exploit any similarity threshold to determine the number of segments appropriate for an image. Such an extension can be added to the last step of hierarchical segmentation.

The segmentation process follows the level sequential algorithm presented in Table 3 until the polygon stabilizes at level L-1. At this point a linking restriction is imposed on the nodes at level L such that only one father node is initially available to the sons at level L-1. All L-1 sons are forced to link to the single father at level L. A total variance for the level L node is calculated and compared to a threshold value. If the variance exceeds the threshold, another father node is made available to the L-1 sons. The nodes at level L are now allowed to link to either of two nodes at the new level. Once level L is stabilized, the variance for each node is calculated and compared to the threshold value. Again, if the variance of any node exceeds the threshold value, another father node at level L is made available to all nodes at level L-1. The additional nodes are introduced as required until either the total variance meets the threshold requirement or until the maximum number of available nodes (4^{N-L}) have been assigned. Once the optimal number of segments has been determined, the segment values are propagated down through the segment subtrees. Only one father is added at a time as opposed to more, e.g., one new father for every old one that exceeded the threshold variance, because the addition of one node typically reduces the variance of many segments, and can bring them to within the threshold.

In this algorithm, the selection of a root segmentation level L becomes less critical as it only defines the maximum number of classes and not the actual number of classes. The actual number of classes is controlled by the variance threshold. The variance threshold, and possibly the segmentation root level L, can be selected dynamically such that segmentation operates closed-loop within the vision system, and in response to the object recognition processing requirements. For example, the variance threshold may be raised in richly textured/colored environments where only gross association is required to accurately group regions. On the other hand, the threshold may be lowered in the situation where fine discrimination is required. This dynamic technique is an implementation of early vision attention allocation.

Figure 7 Segmentation results at root level L=6, n=8 and a variable number (13) of classes.

The segmentation results achieved using a variably sized segmentation level are shown in Figure 7. Comparison of Figures 6c, 6d, and 7 illustrates the advantage of a variably sized segmentation level; the remainder of the segmentation process is identical in all three cases. The 4 classes of Figure 6c are not sufficient to represent individual objects in the scene; distinct objects are merged and lost. The 16 classes of Figure 6d encourage the "over segmentation" of the image; too many closely related segments are not merged. Figure 7 illustrates 13 segments which are the minimum that meet the homogeneity criteria. Note that this number of segments is not a power of two, and so class count is not strictly determined by the hierarchy topology.

Table 4 Dynamic Level Sequential Segmentation Algorithm

1. Perform the Level Sequential Segmentation Algorithm while the processing level k is less than the segmentation level L. When $k = L$-1, go to step 2. Number of segments, or active father cells at level L is ϕ, which is initially set to 1.
2. Each cell at level L-1 has ϕ candidate father cells. Link each cell at level L-1 to its most similarly valued father.

2a. Compute the variance σ_i^2, i=1,...,ϕ for each active father cell using the formula:

$$\sigma_i^2 = \frac{\sum_{j \in s_i} a_j (\mu_j^s - \mu_i^f)^2}{\sum_{j \in s_i} a_j}$$

where s_i is the set of son cells in level L linked to the i^{th} father cell in level L, μ_j^s is the value vector of a son cell, and μ_j^f is the value vector of the i^{th} father cell.

3. If any variance exceeds the variance threshold, increment the number of active fathers ϕ and return to step 2. Otherwise proceed to step 4.
4. Generate the segment values (top-down pass). Propagate down each subtree the value of its root at level L. The segmentation results appear at the cells to which rexel values are assigned (the rexel manifold).

C. Hierarchical Compact Region Detection

Hierarchical segmentation groups pixels or rexels into homogeneous segments that are not necessarily connected. This is illustrated in Figure 8, where the rexels of one segment from Figure 7 are outlined. The 3-D hierarchy is an efficient means of navigating through data and represent groupings of data [Rosenfeld90]. For example, each node in level L efficiently represents a segment, and can readily provide diverse statistical measures on the leaves of the segment without the need of traversing the hierarchy and interrogating each pixel or rexel. Perhaps of greater value are nodes which represent a connected region of a segment. We shall call such a region a homogeneous, compact region (HCR) and define it as a group of leaves belonging to the same segment which is not adjacent to any other leaves of the same segment.

The detection of HCRs represents an important step in the segmentation process. Segments represent homogeneous classes and not necessarily compact regions of interest. This can complicate object detection, which attempts to abstract objects by combining segments. Multiple regions of interest within a segment complicates the interpretation of the segment value vector. For example, the centroid of the segment shown in Figure 8 is found between the two highlighted regions, the

bounding box for the segment encompasses both regions, and yet the segment area accurately describes the expanse of the segment leaves. These values are counterintuitive and undermine object detection performance which depends on accurate segment characterization. A procedure is required which breaks segments into HCRs, which more accurately represent the constituent components of scene objects and thereby facilitate object detection.

An extension to the hierarchical segmentation process was examined and gave very good results. The basic idea is to test each node of the segmented polygon to see if it meets the conditions necessary for it to be flagged as the root of an HCR; the simple conditions are that of homogeneity and compactness. Homogeneity is guaranteed within segmentation subtrees of the stable polygon. Compactness is determined through a two step process of contiguity and adjacency testing.

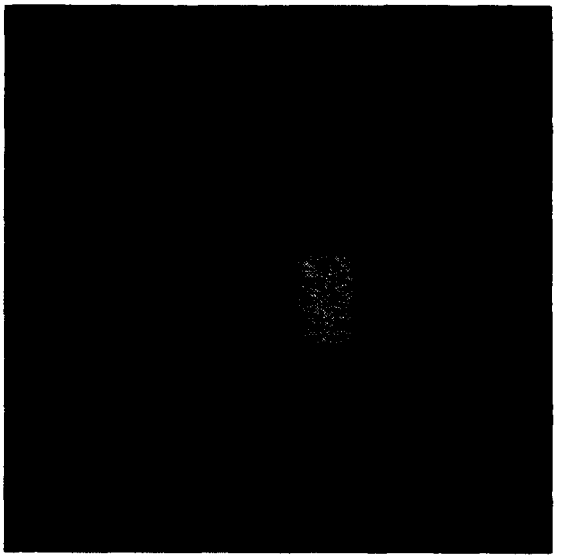
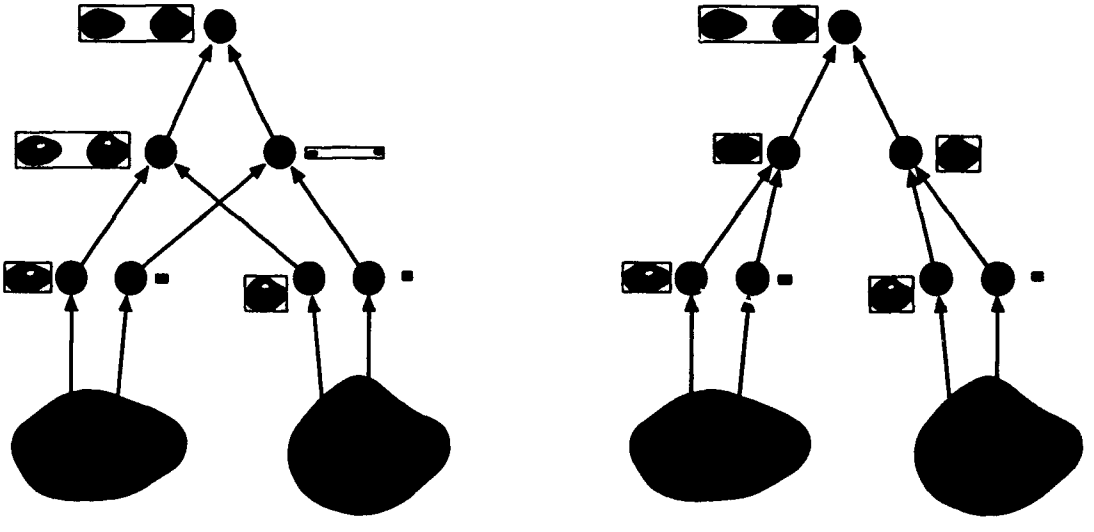


Figure 8 Multiple Compact Regions within a Segment.

A problem occurs when multiple regions within a segment have components which are more similar to themselves than to the rest of the region. These components tend to link to a common father cell which is different than the father cell representing the rest of the regions (Figure 9a). In this case, there is no cell which properly represents a complete object, even though all regions and components are part of the same segment subtree. To correct for this cross-linking, a single bottom-up "corrective pass" is applied to the polygon. In this pass, a cell is relinked to the closest father which is of the same segment. Hence, the shapes of the segments are not altered, and the tree is contiguous (Figure 9b). It should be noted that the result of the corrective pass is more contiguous than if distance had been used as a criteria in the selection of fathers in the initial segmentation process.

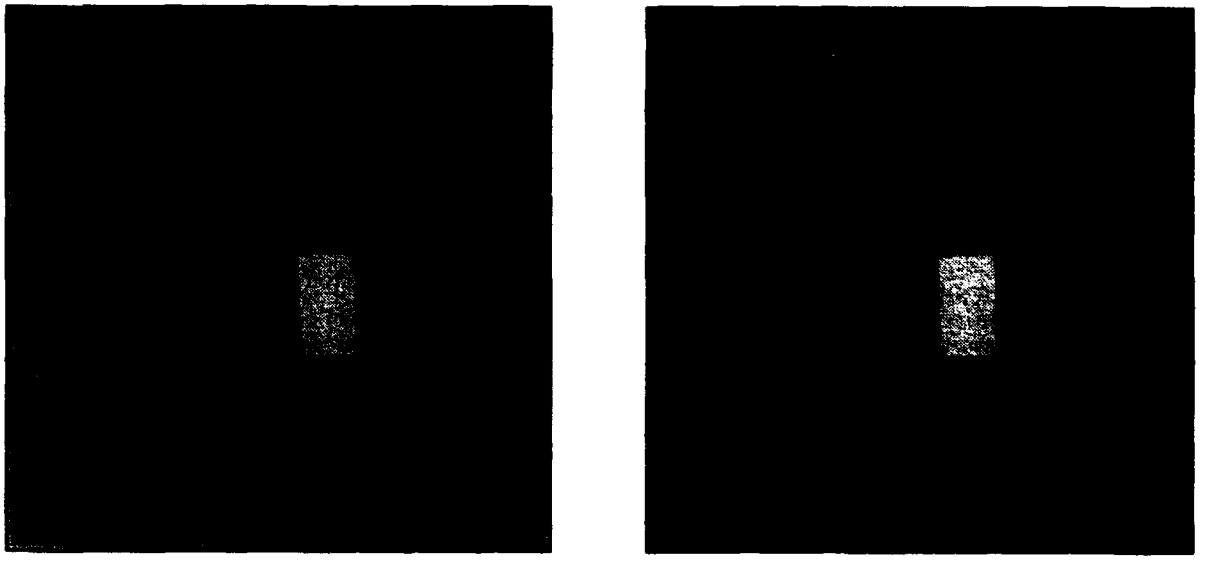


a. Segment Subtree before Corrective Pass

b. Segment Subtree after Corrective Pass

Figure 9 Example of Links and Bounding Boxes in a Segmentation Subtree with Two Objects - Before and After the Corrective Pass

After the corrective pass, the nodes can be tested for adjacency. A single bottom-up pass is performed to identify groups of adjacent nodes within the segment. Leaf cells at level k note their adjacency to neighboring leaf cells of the same segment. The adjacency information from level k is passed to level $k+1$ in the form of an implied adjacency list; it is implied that if two cells at level k are adjacent, their respective father cells are adjacent. Adjacency testing continues through the polygon to the root segment level L. Since the adjacency test is confined to a segment subtree we are guaranteed to converge on a single father cell which is not adjacent to any other cells in its segment. This cell is an HCR root cell. If the HCR root node resides at the root segmentation level L, the HCR subtree is also the segment subtree. If the HCR root node resides below level L, it is one of multiple HCRs within the segment. In either case, the cell is marked as an HCR root node and its value vector is calculated. Since the value vector is an average over the region leaves, it better represents the region of interest than the segment value and thereby facilitates object detection. The algorithm used to performed the adjacency test is provided in Table 5. Figure 10 shows the results of applying the adjacency test to the segment highlighted in Figure 8.



a. First Homogeneous Compact Region. b. Second Homogeneous Compact Region.

Figure 10 Adjacency Test Applied to Figure 8 Segment.

The HCR detection algorithm yields better segmentation results than previous attempts such as thresholding the ratio of each cell's bounding box area to its leaf (subtree) area. The results are not dependent on the selection of a threshold value, nor are they dependent on the selection of the root segmentation level or variance threshold. Since the algorithm does not involve relinking, all contextual information is preserved and readily accessible within the object and segment subtrees.

One additional step has been added to the adjacency based HCR detection algorithm. This step is to correct for the situation where multiple homogeneous, compact regions link to the same HCR root node, illustrated in Figure 11. In Figure 11 the sons on the left comprise one HCR and the sons on the right comprise another HCR and yet both link to the same HCR root node. In this case, the segmentation link structure does not support the separation of HCRs within the segment. In other words too few links are available within the segment subtree to allow each HCR to link to an individual root node. The adjacency test algorithm provided in Table 5 would mislabel the father as

a single HCR. An additional adjacency check is required to ensure that all son cells of an HCR root node are adjacent to each other. Non-adjacency of these son cells implies multiple HCRs.

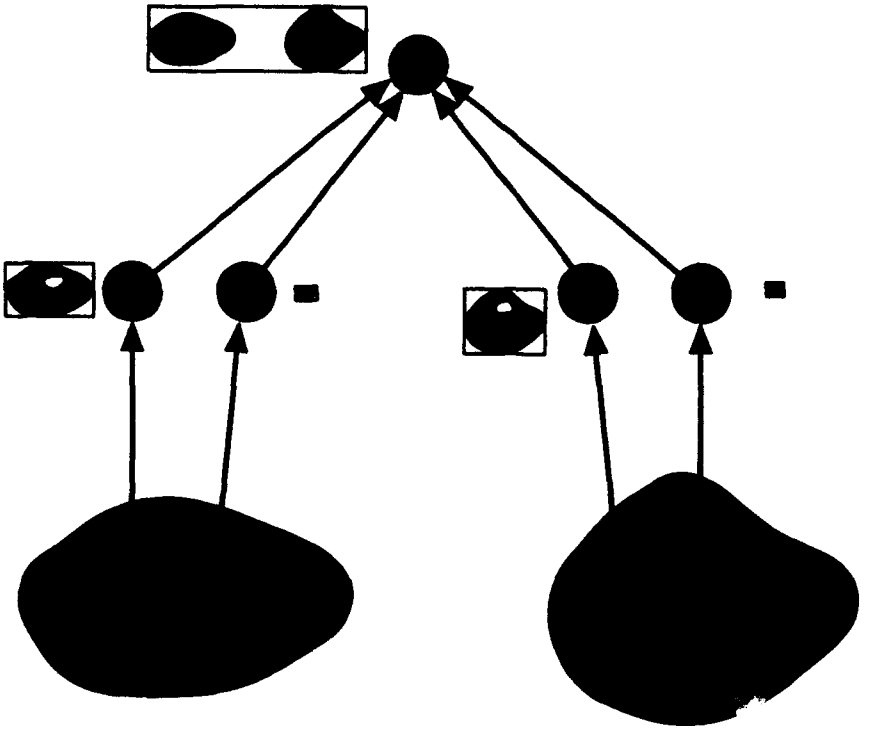


Figure 11 Two HCRs Link to Same Father

Table 5 Adjacency Based HCR Detection Algorithm

1.	Working level k=0. The segmentation value of each cell is compared to the segmentation value of each adjacent cell (3x3). If an adjacent cell has the same segmentation value add the location of that cells father to an implied adjacency list.
1a.	Pass the implied adjacency list to the father of the cell under review. If k=0, advance to step 2.
1b.	If the cell under review is a father cell (level k), access the implied adjacency list passed from its sons (level k-1). Add the location of the father for each implied adjacency cell to the implied adjacency list being generated (level k).
2.	If the cell under review is not adjacent to any other cells in its segment and has a single position value in its adjacency list, mark that cell as an HCR root node. Also, mark the cell as an HCR root node if the cell under review is at the segmentation level L.
2.	Advance to the next level ($k \leftarrow k+1$). HCR detection is complete if k is above the segmentation root level ($k>L$), otherwise return to Step 1.

V. CONCLUSIONS

In this paper we have presented a hierarchical technique to segment foveal images. The technique exploits preliminary, bottom-up, segmentation results to determine the optimal segment count for each image. The segmentation technique also efficiently identifies homogeneous, compact regions (HCR) within each segment. Although HCRs do not necessarily represent objects, the generation of HCRs facilitates object classification efforts by improving the quality of segment statistics. Segment features such as the centroid, distribution variance (zero and first order moments of the segment shape), bounding box of the segment (the smallest box encompassing the segment), and feature variance (first order moment of pixel values) are hierarchically computed for each HCR. These feature values more accurately represent object components than similiar feature statistics calculated for the segment. The technique presented in this paper can easily be adapted to yield similiar results in a traditional image pyramid.

References

1. Cesar Bandera, "Multiacuity target recognition and tracking", Second Automatic Target Recognizer Systems and Technology Conference, DARPA/CNVEO, March 17, 1992.

2. Peter J. Burt, Tsai-Hong Hong, Azriel Rosenfeld, "Image Smoothing Based on Neighbor Linking," IEEE Trans. SMC, vol. 11, no. 12, 1981.

3. Peter J. Burt, "Smart sensing with a pyramid vision machine," Proc. IEEE, vol. 76, no. 8, pp. 1006-1015, 1988.

4. Cibulskis, C. R. Dryer, "Node linking strategies in pyramids for image segmentation," in Multiresolution Image Processing and Analysis, Azriel Rosenfeld (ed.), Springer-Verlag, pp. 109-120, 1984.

5. John A. Hird, David F. Wilson, "A comparison of target detection and segmentation techniques," SPIE Vol. 1191 Optical System for Space and Defense, 1989.

6. Tsai-Hong Hong, K. A. Narayanan, Shmuel Peleg, Azriel Rosenfeld, Teresa Silberberg, "Image Smoothing and Segmentation by Multiresolution Pixel Linking: Further Experiments and Extensions," IEEE Trans. Systems, Man and Cybernetics, vol. 12, no. 5, 1982.

7. Martin D. Levine, Vision in Man and Machine, McGraw Hill, 1985.

8. Shmuel Peleg, Orna Federbusch, Robert Hummel, "Custom-made pyramids," Parallel Computer Vision, Leonard Uhr (ed.), Academic Press, pp. 125-146, 1987.

9. Quentin F. Stout, "Mapping vision algorithms to parallel architectures," Proc. IEEE, vol. 76, no. 8, pp. 982-995, 1988.